UDC 004.89: 510.635 **DOI: 10.20998/2079-0023.2019.02.12**

I. V. LIUTENKO, O. I. KURASOV, D. A. LUKINOVA, S. I. YERSHOVA, A. O. SEMANIK

USING THE AGGREGATED CRITERIA TO EVALUATE THE SOFTWARE TESTS QUALITY

An approach to evaluating the software tests quality using aggregated quality criteria is proposed. The article considers the finding of such characteristics of software tests that can be used to judge their quality and their need for improvement. The subject of the study is the formation of a software tests quality evaluation system, which can be used in the software development process. It is proposed to consider a software test as a multiattribute object. It is emphasized that it is necessary to take into account both quantitative and qualitative characteristics of tests and test coverage, which greatly complicates the construction of a model for evaluating the software tests quality. Various approaches to solving the problem of evaluating complex, multiattribute objects are considered. The problem of comparing and ordering complex objects taking into account different criteria is considered. The choice of the method of sequential aggregation of classified states to solve the problem of multi-criteria selection and assessment is justified. The stages of the procedure for solving the estimation problem using the method of sequential aggregation of classified states are considered. An activity diagram is constructed that reflects an algorithm for constructing a hierarchical system of criteria. The criteria for evaluating software tests are given, which belong to three groups - efficiency, coverage, and software implementation. For a hierarchical system of criteria aggregation, a set of indicators, their qualitative gradations with corresponding numerical intervals, are allocated. At the highest level of the hierarchy, it is proposed to use three composite criteria that correspond to the groups of efficiency, coverage and implementation, which will allow to obtain an integral indicator of the software tests quality. The resulting integral indicator includes five quality classes, each of which corresponds to a multitude of low-level indicator estimates. Tests quality evaluation will improve the testing process, which purpose is to ensure the specified quality of the software being developed.

Keywords: software, testing, quality, evaluation, assessment criteria, multiattribute object, aggregated criterion.

І. В. ЛЮТЕНКО, О. І. КУРАСОВ, Д. А. ЛУКІНОВА, С. І. ЄРШОВА, А. О. СЕМАНИК

ВИКОРИСТАННЯ АГРЕГОВАНИХ КРИТЕРІЇВ ДЛЯ ОЦІНКИ ЯКОСТІ ТЕСТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Пропонується підхід до оцінки якості тестів програмного забезпечення з використанням агрегованих критеріїв якості. Розглядається знаходження таких характеристик тестів програмного забезпечення, за якими можна судити про їхню якість і необхідність доопрацювання. Предметом дослідження є формування системи оцінювання якості програмних тестів, яку можливо використовувати в процесі розробки програмного забезпечення. Запропоновано розглядати тест програмного забезпечення як багатоознаковий об'єкт. Підкреслюється, що необхідно враховувати як кількісні, так і якісні характеристики тестів і тестового покриття, що істотно ускладнює побудову моделі оцінки якості програмних тестів. Розглянуто різні підходи до вирішення задачі оцінювання складних, багатоознакових об'єктів. Розглядається проблема порівняння й упорядкування складних об'єктів з урахуванням різних критеріїв. Обґрунтовано вибір методу послідовного агрегування станів, що класифікуються для розв'язання задачі багатокритеріального вибору і проведення оцінювання. Розглянуто етапи процедури вирішення задачі оцінювання з використанням методу послідовного агрегування станів, що класифікуються. Наведена діаграма діяльності, яка відображає алгоритм побудови ієрархічної системи критеріїв. Розглянуті критерії оцінювання програмних тестів, які відносяться до трьох груп – ефективності, покриття і програмної реалізації. Для ієрархічної системи агрегування критеріїв виділено набір показників, їх якісні градації з відповідними чисельними інтервалами. На вищому рівні ієрархії запропоновано використовувати три складених критерія, які відповідають групам ефективності, покриття і реалізації, що, в свою чергу, дозволить отримати інтегральний показник якості програмних тестів. Отриманий інтегральний показник включає п'ять класів якості, кожному з яких відповідає множина оцінок показників нижнього рівня. Оцінка якості програмних тестів дозволить поліпшити процес тестування, метою якого є забезпечення заданого рівня якості програмного забезпечення, що розробляється.

Ключові слова: програмне забезпечення, тестування, якість, оцінювання, критерії оцінки, багатоознаковий об’єкт, агрегований критерій.

И. В. ЛЮТЕНКО, А. И. КУРАСОВ, Д. А. ЛУКИНОВА, С. И. ЕРШОВА, А. А. СЕМАНИК

ИСПОЛЬЗОВАНИЕ АГРЕГИРОВАННЫХ КРИТЕРИЕВ ДЛЯ ОЦЕНКИ КАЧЕСТВА ТЕСТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Предлагается подход к оценке качества тестов программного обеспечения с использованием агрегированных критериев качества. Рассматривается нахождение таких характеристик тестов программного обеспечения, по которым можно судить об их качестве и необходимости доработки. Предметом исследования является формирование системы оценивания качества программных тестов, которую возможно использовать в процессе разработки программного обеспечения. Предложено рассматривать тест программного обеспечения как многопризнаковый объект. Подчеркивается, что необходимо учитывать как количественные, так и качественные характеристики тестов и тестового покрытия, что существенно усложняет построение модели оценки качества программных тестов. Рассмотрены различные подходы к решению задачи оценивания сложных, многопризнаковых объектов. Рассматривается проблема сравнения и упорядочения сложных объектов с учетом разных критериев. Обоснован выбор метода последовательного агрегирования классифицируемых состояний для решения задачи многокритериального выбора и проведения оценивания. Рассмотрены этапы процедуры решения задачи оценивания с использованием метода последовательного агрегирования классифицируемых состояний. Приведена диаграмма деятельности, которая отражает алгоритм построения иерархической системы критериев. Рассмотрены критерии оценивания программных тестов, которые относятся к трем группам – эффективности, покрытия и программной реализации. Для иерархической системы агрегирования критериев выделен набор показателей, их качественные градации с соответствующими численными интервалами. На высшем уровне иерархии предложено использовать три составных критерия, которые соответствуют группам эффективности, покрытия и реализации, что, в свою очередь, позволит получить интегральный показатель качества программных тестов. Полученный интегральный показатель включает пять классов качества, каждому из которых соответствует множество оценок показателей нижнего уровня. Оценка качества тестов позволит улучшить процесс тестирования, целью которого является обеспечение заданного качества разрабатываемого программного обеспечения.

Ключевые слова: программное обеспечение, тестирование, качество, оценивание, критерии оценки, многопризнаковый объект, агрегированный критерий.

Introduction. Much of modern software (SW) is a complex, multi-component system with a large amount of software code, which can include a wide range of components that perform a variety of tasks. A list of functional and non-functional requirements is advanced to the software systems themselves (SS), which complex programming logic is often implemented for, which must work with special conditions and restrictions.

The complexity of this task makes software testing an important step in the development of the oftware systems of any type and scope. First, it prevents and corrects defects which make it impossible to use the application, that again keeps users out of achieving their own goal. Secondly, testing is necessary to verify the compliance of the software product with the requirements that have been put forward by the customer and stakeholders.

The right approach to testing will allow to supply the customer with a quality product, but this requires a responsible approach to the organization of testing, design and development of software tests. Software test quality evaluation will provide an opportunity to create such a complex of tests for various purposes, which will allow to control the quality of the software with the least expenses for testing.

Formulation of the problem. The purpose of the study is to define the criteria for assessing the quality of tests, which will allow to exclude the subjectivity of the expert. The relevance of the work is due to the fact that software testing, as well as the other stages of software development, is performed in conditions of limited time and financial resources. This means that voluminous and detailed testing of the entire SS is unprofitable and sometimes impossible. Requirements, software components differ in priority and complexity, which can be expressed in quantitative and qualitative terms.

This means that the priority and complexity of the test object (in this case an individual component of the SS or requirement) requires an appropriate amount of software tests. A test group that was formed without these characteristics cannot be considered qualitative, because incorrectly defined testing priorities lead to waste of time and cost, which is not guaranteed by the sufficient reliability of the SS that was released after such tests. The purpose of the study is to find indicators that can determine the value and usefulness of the software tests that are offered for software testing.

The problem of multi-criteria selection is formed as follows. There are many options $A_1, \dots, A_p$, each of which is characterized by specific criteria $K_1, \dots, K_m$. Each criterion K_i has a scale $X_i = \{x_i^1, \dots, x_i^{g_i}\}, i = 1,..m$, which has in most cases ordered discrete numeric or verbal gradations. It is necessary, based on the preferences of the decision maker, to choose one or more of the best options from the set presented.

The main difficulty is that both quantitative and qualitative indicators of varying degrees of importance need to be analyzed, many of which greatly complicate the comparison of tests and test coverage. An additional fact is that there is no single quality assessment model to evaluate the quality of the tests.

In the case of evaluating many objects with several dozen properties, there is a problem that comparing only one attribute value becomes impossible, and attempts to reduce the number of evaluation criteria leads to a decrease in the quality of the final result by pulling it away from the reality. Such conditions require finding a method that would solve the problem of multicriteria selection in the large space by reducing the number of dimensions, based on the rules of the subject area and the specifics of the objects being compared. The reduction in the number of measurements will be used to aggregate multiple criteria to one to obtain a grading scale that depends on the preferences of the decision maker (DM).

Existing methods for solving the problem. The solution to a similar problem can be constructed on the basis of the problem of finding the extremum of one or more utility (value) functions [1]. To complete the task, it is necessary to derive a generalized criterion from many numerical criteria by minimizing them and finding a weighted sum. With a large number of criteria, this method is too time-consuming because it requires the domain analyst to spend a great deal of time in deriving the approximate utility function, as well as the importance factors (weights) that must be assigned to each property taken into account, which in itself is a task of ambiguous solution. Another disadvantage of this method is that the use of aggregated indicators does not allow you to reproduce the input data, which implies the inability to easily explain the results of the comparison [2]. The use of coarse sets in the classification of multi-criteria objects is to use the sets of rules defined by DM to classify alternatives to a particular class with varying degrees of accuracy. The method is complex enough because a large number of classification rules complicates their analysis. In addition, the method requires pre-debugging on the prepared data sets [3].

Often, methods are based on pair-wise comparisons of objects to organize objects as a whole or by many criteria. Complete ordering of objects occurs when you can compare all pairs of variants and DM preferences are transitive. If some of the pairs cannot be compared, partial ordering will be obtained. In methods of analytical hierarchy [4] variants are ordered according to their priority index, which is consistently calculated by pairwise comparison of variants, criteria of their evaluation and participants in relation to the global goal of the problem being solved. The disadvantage is the sensitivity to the context of the choice, which leads to a dramatic change in ordering after adding / excluding a particular variant. In [5] there are two main methods of comparison: the first is the direct sorting of objects by given classes, which is the most popular method of classification due to ease of use, and the second is an interactive classification procedure that provides a description of DM preferences through the utility function, which is weighted sum of many scalar criteria.

Given the poor structure of the problem, the methodology of verbal analysis of solutions can be used. According to this methodology, the properties of variants

are described using qualitative criteria that have verbal formulations of gradations on the rating scales [6].

To solve the problem of multicriteria selection, the "PAKS" method (sequential aggregation of classified states) was selected. This method is characterized by the use of verbal analysis methods to reduce the dimension space of the criteria. The method was chosen because hierarchical evaluation of complex qualitative criteria will allow to obtain meaningful and understandable evaluation with the least time spent on building an evaluation system for DM [7]. The "PAKS" decision procedure has three steps.

The first step is to build a hierarchical system of aggregated criteria by "ISKRA" (hierarchical convolution of criteria and attributes) taking into account the beliefs of DM. The process of construction is to create integral indicators that characterize the properties of options that are selected based on domain concepts, which aggregate the initial characteristics. The procedure for aggregation of indicators is consistent, that is, the obtained sets of criteria are grouped in series into new groups of the next level of the hierarchy, and so on up to a single integral criterion of the highest level, if necessary.

In the second step, the sequential classification task performs a consistent scale construction of each composite criterion, which consists of using a combination of grading estimates of the output indicators as classified objects. Classes are graded scores of the composite criterion, so that every combination of gradations of the original scores will match some gradation of scores on the composite criterion scale [8]. In the general case, virtually any method of ranking or classification of multi-criteria alternatives can be used to construct scales of composite criteria, which allows to present each gradation of the composite criterion scale in the form of a combination of gradations of baseline scores.

In the third stage, the final solution of the problem of selection in the obtained space of complex criteria of smaller dimension using the method of "ARAMIS" (aggregation and ranking of alternatives to multipurpose ideal situations) [9], which allows to rank objects described by many periodic quantitative and / or qualitative attributes $K_1, \dots, K_m$, without constructing individual object rankings. Multi-criteria objects $A_1, \dots, A_p$ are considered as points of a metric space of multisets with some metric, which are compared and ordered in terms of relative proximity to the best (ideal) object A_+ or worst (anti-ideal) A_- in that space. The best and worst objects (which may also be hypothetical) have the highest and lowest scores by all criteria, respectively. All objects are ordered by proximity to the best object A_+, by distance from the worst object A_- or by the value of relative proximity to the best object.

Fig. 1 shows a diagram of the activity of solving the multicriteria selection problem with a consistent reduction in the dimension of the feature space.

To obtain a comprehensive assessment of the test quality, it is necessary to consider a large number of criteria that can be attributed to the groups of efficiency, coverage and software implementation

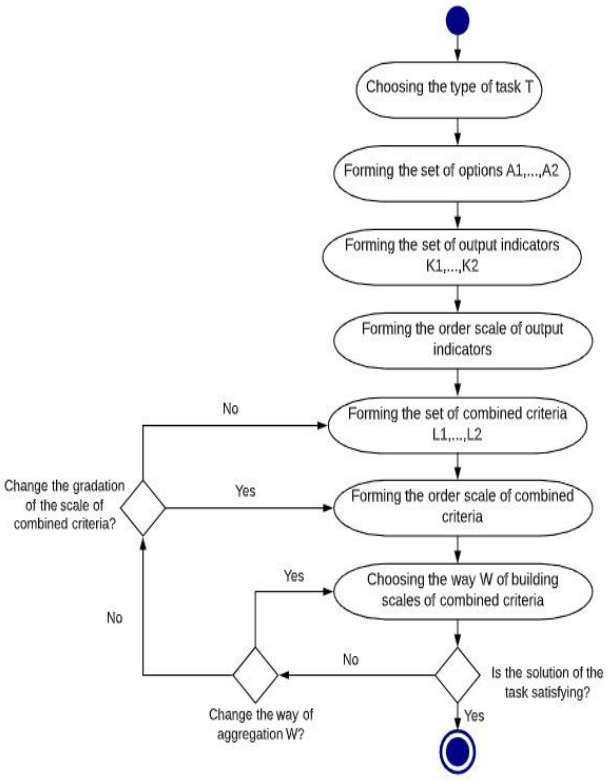

Fig. 1. Diagram of the algorithm for building a hierarchical system of criteria

On the other hand, to cover a larger volume of software projects requires a set of criteria with which the evaluation process remains relevant to the specifics of the individual project. Creating higher level criteria allows you to create new, more general levels of composite criteria by reducing the number of combinations of grading criteria. Detailed metrics can be grouped in their essence, making it possible to obtain an integrated Quality Score. For numeric metrics, you can enter qualitative gradations that match a certain range of values. Practice shows that the success of software testing depends on the quality of test planning and implementation. Testing performance can be estimated from a relatively small number of indicators.

The first indicator is the ratio of not intercepted in the latest software version bugs to the number of bugs found (found and fixed / not intercepted). This indicator may characterize the thorough testing of different use cases of SS. Completely covering all variants of data, conditions and actions is an almost impossible task, so there is a risk that the user may perform an unchecked sequence of actions that will disrupt the normal operation of the software. Finding data that has not yet been intercepted should be accompanied by adjusting program logic and introducing new warning tests, which will help to reduce the ratio. It is worth noting that bugs found at the testing stage for various reasons may not be documented and, in turn, not corrected, which makes it untouched.

This indicator in percentage terms can be calculated by the following formula:

$$E_1 = \frac{N}{M} \cdot 100, \quad (1)$$

where E_1 – the ratio not intercepted in the latest software version bugs to the number of all bugs found;

N – the current number of bugs not intercepted after testing;

M – the total number of bugs that have been detected since the last test started.

The second indicator is the proportion of bugs that were repeated in the release – these bugs were fixed in previous versions, but became relevant again after the release and the first use the new software version. This indicator differs from the previous one in that it may indicate a problem of lack of regression testing, while the first indicator is more relevant for determining the quality of functional testing introduced in the latest version. The indicator can be calculated by the following formula:

$$E_2 = \frac{N_r}{M_r} \cdot 100, \quad (2)$$

where E_2 – the number of bugs that were repeated in the release;

N_r – the number of repeated bugs;

M_r – the total number of fixed bugs.

The disadvantage is the complexity of the calculation due to the existence of system dependencies of the new and previously developed program code, which makes possible the situation when the new functionality doesn't work due to previously found defects in the old one [10].

The amount of functionality coverage should show how comprehensively the capabilities of the software have been tested. For each project, you can determine your own, satisfactory coverage level. The metric can be calculated as the ratio of the number of opportunities tested to the total number of opportunities.

The total number of functional requirements covered can be calculated using the requirements trace matrix. In the simplest form, this matrix is a table on the rows of which the functional requirements for the application are placed, and on columns the test scenarios. In the special circumstances of the project, columns and rows with additional information may be added. Related features and scripts should be marked at the intersection of the row and column, so that testers get clear information about the current coverage. One test scenario for one function is considered sufficient coverage, so it is necessary to break down the complex requirements into atomic components. This approach simplifies the analysis of congestion or lack of tests [11].

Also, when evaluating a test coverage, a feature coverage indicator can be used, which is calculated as the ratio of the number of features tested to the total number of features. For the needs of a particular project, those functions that represent complex operations of an application's business logic can be included in the list of functions.

Most software tests are software-implemented, which makes it possible to evaluate them as a separate software system with its own interconnected components. When evaluating tests as code, you can use the following code properties:.

1) compliance with the rules of programming language (conventions) - this indicator affects the ease of perception of the program code, which is important when accompanied by several developers;

2) code purity - the structural simplicity of the code (for example, the adequate amount of method or class), the absence of unnecessary structures (imports, variables) left after the code is modified or refactored, as well as those structures that interfere with code maintenance and analysis ("Magic numbers", duplicates) [12]. These metrics can be measured as the volume of violations per 1000 lines of test code.

Table 1 – Test evaluation criteria

Indicator	Qualitative grading / order of stickiness	Interval
1. Amount of not intercepted bugs (%)	High / 2	(80;100]
	Middle / 1	(30;80]
	Low / 0	[0;30]
2. Amount of returned bugs (%)	High / 2	[50;100]
	Middle / 1	(20;50)
	Low / 0	[0;20]
3. Test coverage of capabilities (%)	High / 0	(60;100]
	Middle / 1	(20;60]
	Low / 2	[0;20]
4. Test coverage of software features (%)	High / 0	(60;100]
	Middle / 1	(20;60]
	Low / 2	[0;20]
5. Compliance with programming language standards (violations per thousand pages of code)	High / 0	[0;10]
	Middle / 1	[10;20]
	Low / 2	More then 20
6. Purity of code (violations per thousand pages of code)	High / 0	[0;5]
	Middle / 1	[5;15]
	Low / 2	More then 15

Simulation results. Table 1 lists the main evaluation criteria, their qualitative gradations, together with the corresponding numerical intervals.

The order of stickiness is given in ascending order (0 is the best, 1 is satisfactory, 2 is bad) and is used for two levels of the hierarchy.

It was proposed to use three composite criteria at the top level of the hierarchy – efficiency, coverage and implementation.

The performance criterion included the amount of bugs not intercepted (%) and the number of bugs returned (%).

The coverage criterion included test capability coverage (%) and test feature coverage (%).

The implementation criterion included compliance with the rules of programming language (violations per thousand pages of code) and purity of code (violations per thousand pages of code).

Table 2 lists the gradations of the aggregated criteria and the corresponding tuples of the graded subordinate criteria.

Table 2 – Composite test quality criteria

Criterion	Gradation / order of stickiness	Corteges of child estimates
Efficiency	High / 0	<0;0>,<0;1>,<1;0>,<2;0>, <0;2>
	Middle / 1	<1;2>,<2;1>,<1;1>
	Low / 2	<2;2>
Coverage	High / 0	<0;0>,<0;1>,<1;0>,<0;2>
	Middle / 1	<1;2>,<1;1>,<2;0>
	Low / 2	<2;2>, <2;1>
Realizatio n	Good / 0	<0,0>;<0,1>;<1,0>
	Satisfactory / 1	<1,2>;<2,1>;<1,1>;<2,0>; <0,2>
	Needs adjustments/2	<2,2>

Integral Quality Score can be represented as five consecutive quality classes, each of which corresponds to a set of tuples of second-level metrics (<Performance, Coverage, Realization>).

The first class corresponds to <0; 0; 0>.

The second class corresponds to <0; 0; 1>, <0; 0; 2>, <0; 1; 0>, <0; 1; 1>, <1; 0; 0>, <1; 0; 1> , <1; 0; 2>.

The third class corresponds to <0; 1; 2>, <0; 2; 1>, <0; 2; 2>, <0; 2; 0>, <1; 1; 0>, <1; 1; 1> , <1; 1; 2>, <1,2,0>.

The fourth class corresponds to <2; 0; 1>, <2,0,0>, <1,2,1>, <1,2,2>, <2; 0; 2>, <2; 1; 0> .

The fifth grade corresponds to <2; 2; 2>, <2; 2; 1>, <2; 2; 0>, <2; 1; 2>, <2; 1; 1>.

Conclusions. Developing an approach for software tests quality evaluation can in the long term improve test results, reduce the time and other resources spent on finding defects in the software system and quickly eliminate the shortcomings of the current testing approach. The obtained results confirm the possibility to use the indicators that can be used to evaluate the overall quality of software tests. These include test performance metrics, test coverage of software capabilities and its software code, namely functions, as well as metrics that make it feasible to use the tests themselves. For these criteria, metrics were formed, the intervals of which were defined as qualitative indicators, which were used to create a hierarchical system of criteria that allows to obtain an integral quality index.

References

1. Петровский А. Б. *Теория принятия решений*. Москва: Издательский центр «Академия», 2009. 398 с.
2. Саати Т. *Принятие решений. Метод анализа иерархий*. Москва: Радио и связь, 1993. 278 с.
3. Doumpos M., Zopounidis C. *Multicriteria Decision Aid Classification Methods*. Dordrecht: Kluwer Academic Publishers, 2002. 245 p.
4. Köksalan M., Ulu C. An interactive approach for placing alternatives in preference classes. *European Journal of Operational Research*. 2003. Vol. 144, no. 2, pp. 429–439.
5. Ларичев О. И. *Вербальный анализ решений*. Москва: Наука, 2006. 181 с.
6. Ройзензон Г. В. Способы снижения размерности признакового пространства для описания сложных систем в задачах принятия решений. *Новости искусственного интеллекта*. 2005. № 1. С. 18–28.
7. Петровский А. Б., Ройзензон Г. В. Многокритериальный выбор с уменьшением размерности пространства признаков: многоэтапная технология ПАКС. *Искусственный интеллект и принятие решений.* 2012. № 4. С. 88–103.
8. Фуремс Е. М. Модифицированный метод экспертной номинально-порядковой классификации. *Искусственный интеллект и принятие решений*. 2010. № 4. С. 81–93.
9. Петровский А. Б., Тихонов И. П. Фундаментальные исследования, ориентированные на практический результат: подходы к оценке эффективности. *Вестник РАН*. 2009. Т. 79. № 11. С. 1006–1011.
10. *Important Software Test Metrics and Measurements*. URL: http://www.softwaretestinghelp.com/software-test-metrics-and-measurements (access date: 23.01.2019).
11. Gotel O., Cleland-Huang J., Hayes, J., Zisman A., Egyed A. *Software and Systems Traceability*. London: Springer, 2012. 152 p.
12. *A SLOC Counting Standard*. URL: http://csse.usc.edu/TECHRPTS/2007/usc-csse-2007-737/usc-csse-2007-737.pdf (access date: 03.06.2019).

References (transliterated)

1. Petrovskiy A. B. *Teoriya prinyatiya resheniy* [The decision theory]. Moscow, "Akademiya" Publ., 2009. 398 p.
2. Saaty T. *Prinyatie resheniy. Metod analiza iererhiy* [The making decisions. Hierarchy analysis method]. Moscow, Radio i svyaz Publ., 1993. 278 p.
3. Doumpos M., Zopounidis C. *Multicriteria Decision Aid Classification Methods*. Dordrecht: Kluwer Academic Publishers, 2002. 245 p.
4. Köksalan M., Ulu C. An interactive approach for placing alternatives in preference classes. *European Journal of Operational Research*. 2003. Vol. 144, no. 2, pp. 429–439.
5. Larichev O. I. *Verbalnyiy analiz resheniy* [The verbal decision analysis]. Moscow, Nauka Publ. 2006. 181 p.
6. Royzenzon G. V. Sposobyi snizheniya razmernosti priznakovogo prostranstva dlya opisaniya slozhnyih sistem v zadachah prinyatiya resheniy [Ways to reduce the dimension of feature space for describing complex systems in decision-making problems]. *Novosti iskusstvennogo intellekta* [Artificial Intelligence News]. 2005, no. 1, pp. 18–28.
7. Petrovskiy A. B., Royzenzon G. V. Mnogokriterialnyiy vyibor s umensheniem razmernosti prostranstva priznakov: mnogoetapnaya tehnologiya PAKS [Multi-criteria selection with reduced dimension of feature space: multi-stage PAKS technology]. *Iskusstvennyiy intellekt i prinyatie resheniy* [Artificial Intelligence and Decision Making]. 2012, no. 4, pp. 88–103.
8. Furms Y. M. Modifitsirovannyiy metod ekspertnoy nominalno-poryadkovoy klassifikatsii [The modified method of expert nominal ordinal classification]. *Iskusstvennyiy intellekt i prinyatie resheniy* [Artificial Intelligence and Decision Making]. 2010, no. 4, pp. 81–93.

9. Petrovskiy A. B., Tihonov I. P. Fundamentalnyie issledovaniya, orientirovannyie na prakticheskiy rezultat: podhodyi k otsenke effektivnosti [Result-oriented basic research: approaches to evaluating effectiveness]. *Vestnik RAN* [RAS Bulletin]. 2009, vol. 79, no. 11, pp. 1006–1011.
10. *Important Software Test Metrics and Measurements.* URL: http://www.softwaretestinghelp.com/software-test-metrics-and-measurements (access date: 23.01.2019).
11. Gotel O., Cleland-Huang J., Hayes, J., Zisman A., Egyed A. *Software and Systems Traceability*. London: Springer, 2012. 152 p.
12. *A SLOC Counting Standard*. URL: http://csse.usc.edu/TECHRPTS/2007/usc-csse-2007-737/usc-csse-2007-737.pdf (access date: 03.06.2019).

Received 05.09.2018

Відомості про авторів /Сведения об авторах/ About the Authors

Лютенко Ірина Вікторівна – кандидат технічних наук, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри програмної інженерії та інформаційних технологій управління; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-4357-1826; e-mail: liv@kpi.kharkov.ua

Курасов Олексій Ігорович – Національний технічний університет «Харківський політехнічний інститут», студент; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-2518-577X; e-mail: kurasov.oleksii@gmail.com

Лукінова Дарина Андріївна – фізична особа-підприємець; м. Харків, Україна; ORCID: https://orcid.org/0000-0002-3644-9972; e-mail: dasha.lutenko@gmail.com

Єршова Світлана Іванівна – Національний технічний університет «Харківський політехнічний інститут», старший викладач кафедри програмної інженерії та інформаційних технологій управління; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-3893-117X; e-mail: svetlana.ershova.2016@gmail.com

Семаник Анастасія Олександрівна – Національний технічний університет «Харківський політехнічний інститут», студентка; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-0653-5359; e-mail: bilenko.anastasiia1@gmail.com

Лютенко Ирина Викторовна – кандидат технических наук, Национальный технический университет «Харьковский политехнический институт», доцент кафедры программной инженерии и информационных технологий управления; г. Харьков, Украина; ORCID: https://orcid.org/0000-0003-4357-1826; e-mail: liv@kpi.kharkov.ua

Курасов Алексей Игоревеч – Национальный технический университет «Харьковский политехнический институт», студент; г. Харьков, Украина; ORCID: https://orcid.org/0000-0003-2518-577X; e-mail: kurasov.oleksii@gmail.com

Лукинова Дарина Андреевна – физическое лицо-предприниматель; г. Харьков, Украина; ORCID: https://orcid.org/0000-0002-3644-9972; e-mail: dasha.lutenko@gmail.com

Ершова Светлана Ивановна – Национальный технический университет «Харьковский политехнический институт», старший преподаватель кафедры программной инженерии и информационных технологий управления; г. Харьков, Украина; ORCID: https://orcid.org/0000-0003-3893-117X; e-mail: svetlana.ershova.2016@gmail.com

Семаник Анастасия Александровна – Национальный технический университет «Харьковский политехнический институт», студентка; г. Харьков, Украина; ORCID: https://orcid.org/0000-0003-0653-5359 ; e-mail: bilenko.anastasiia1@gmail.com

Liutenko Iryna Victorivna – Candidate of Engineering Sciences, National Technical University "Kharkiv Polytechnic Institute", Associate Professor, Department of Software Engineering and Management Information Technology; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-4357-1826; e-mail: liv@kpi.kharkov.ua

Kurasov Oleksii Igorovych – National Technical University "Kharkiv Polytechnic Institute", student; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-2518-577X; e-mail: kurasov.oleksii@gmail.com

Lukinova Daryna Andriivna – entepreneur; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0002-3644-9972; e-mail: dasha.lutenko@gmail.com

Yershova Svitlana Ivanivna – National Technical University "Kharkiv Polytechnic Institute", Senior Lecturer in Department of Software Engineering and Management Information Technology; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-3893-117X; e-mail: svetlana.ershova.2016@gmail.com

Semanyk Anastasiia Oleksandrivna – National Technical University "Kharkiv Polytechnic Institute", student; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-0653-5359 ; e-mail: bilenko.anastasiia1@gmail.com